DOI: 10.1111/cgf.14687
Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 41 (2022), Number 7

EL-GAN: Edge-Enhanced Generative Adversarial Network for Layout-to-Image Generation

Lin Gao, Lei Wu† and Xiangxu Meng

School of Software, Shandong University, Jinan, Shandong, China

Abstract

Although some progress has been made in the layout-to-image generation of complex scenes with multiple objects, object-level generation still suffers from distortion and poor recognizability. We argue that this is caused by the lack of feature encodings for edge information during image generation. In order to solve these limitations, we propose a novel edge-enhanced Generative Adversarial Network for layout-to-image generation (termed EL-GAN). The feature encodings of edge information are learned from the multi-level features output by the generator and iteratively optimized along the generator's pipeline. Two new components are included at each generator level to enable multi-scale learning. Specifically, one is the edge generation module (EGM), which is responsible for converting the output of the multi-level features by the generator into images of different scales and extracting their edge maps. The other is the edge fusion module (EFM), which integrates the feature encodings refined from the edge maps into the subsequent image generation process by modulating the parameters in the normalization layers. Meanwhile, the discriminator is fed with frequency-sensitive image features, which greatly enhances the generation quality of the image's high-frequency edge contours and low-frequency regions. Extensive experiments show that EL-GAN outperforms the state-of-the-art methods on the COCO-Stuff and Visual Genome datasets. Our source code is available at https://github.com/Azure616/EL-GAN.

CCS Concepts

• ***Computing methodologies*** → *Scene understanding; Image processing;*

1. Introduction

Tremendous progress has been made in the conditional image generation task [ZMYS19, QZXT19b, JGFF18, LWTT18, NFLW20] that aims to generate plausible images based on user-specified conditions. In this paper, we pay attention to image generation from layout. By specifying a layout (including bounding boxes and corresponding object/stuff category labels), an image that conforms to it can be generated. From an application point of view, the task of generating images from layouts can quickly turn a virtual composition in mind into an actual image, enabling everyone to become an artist, and has excellent prospects for development.

Image generation from layout is a relatively new and challenging task. Layout2Im [ZMYS19] took the lead in proposing this task. However, the generated images are low resolution, and some generated objects are difficult to identify. Attribute-guided layout2im [MZS20] allows (but does not require) direct semantic attribute control over individual objects. In comparison, OC-GAN [SZB*20] restricts the relationship between objects by introducing scene graphs as an additional condition. The fine-grained control of objects in the image has gradually attracted researchers' attention. Nevertheless, existing methods still suffer from distortion, ambiguity, and poor recognizability in an object-level generation. As shown in the second column of Figure 1, the head of the *sheep* generated by LostGAN-V2 [SW20] is missing, as are the *giraffe's* legs. Moreover, the *train* has undergone severe deformation.

The human visual system is susceptible to high-frequency edge information, and the edge contour of an object largely determines the human perception of it. An important reason for the above problems in the object-level generation is that the quality of the edge lines of the object is inferior. Just imagine, if the edge lines of a "car" in the picture are clear and smooth, people will not pay too much attention to details such as its texture but directly determine that it is a car by the edge contour of the object. Conversely, if the edge lines of a "car" in the image are distorted, even if its texture resembles a car, people will find it strange or difficult to recognize because such a car does not exist in the real world. So for the image generation task, we confirm that the feature encodings for edge information can be added to the generator while feeding frequency-sensitive image features to the discriminator to solve the problem of object-level generation.

Therefore, we propose a novel Generative Adversarial Network

† Corresponding author: i_lily@sdu.edu.cn

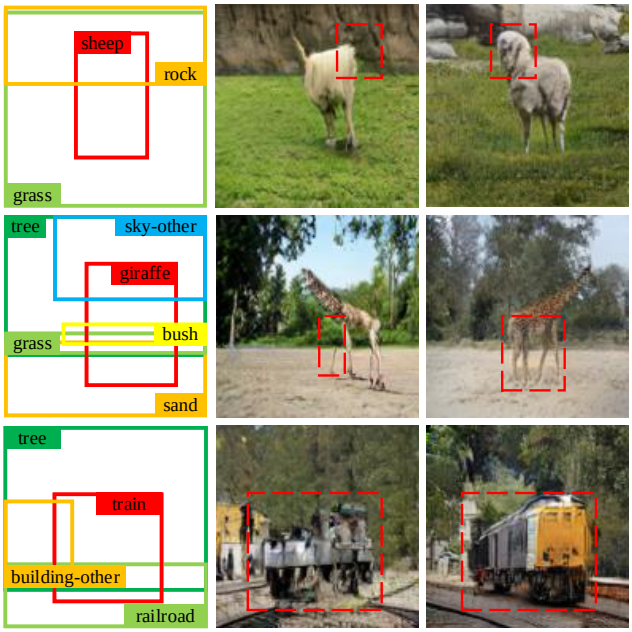

Figure 1: *The performance of existing methods and our method on the edge processing problem. Regions with significant differences in generation quality are marked with red dashed rectangles. It can be seen from the comparison that the edge contour of each component in the image generated by our model is more reasonable, and the object recognition is higher.*

architecture with enhanced **E**dge information for **L**ayout-based image generation, called EL-GAN for short. The feature encodings containing edge information are learned from the multi-level features output by the generator and iteratively optimized along the generator's pipeline. Two new components are linked at each generator level to realize the extraction and fusion of edge information, called edge generation module (EGM) and edge fusion module (EFM). EGM converts the output of the multi-level features by the generator into edge maps of different scales in two stages to facilitate multi-scale training. Specifically, multi-level features are first transformed into intermediate image results at different scales through "ToImg," and then the edge maps of these images are extracted via "GetEdge." Next, we use EFM to effectively fuse the feature encodings refined from the edge maps into the subsequent image generation process by modulating the affine transformation parameters in the normalization layers. At the same time, we transform the image from the spatial domain to the frequency domain, and the frequency representations of these images are fed to the discriminator. The discriminator is required to score the authenticity of the image through frequency-sensitive image features, which enhances the generation quality of high-frequency edge contours and low-frequency regions of the image. These new modules work together to help generate smooth and full edge lines, which significantly alleviates the problem of object deformation and improves object recognizability (see the third column of Figure 1). The main contributions of our work are as follows:

- We propose a novel edge-enhanced Generative Adversarial Network for layout-to-image generation (termed EL-GAN). Through multi-scale learning, it can generate images with clearer edge lines and better recognizable objects based on layout conditions.
- Two new components are proposed, called edge generation module (EGM) and edge fusion module (EFM). They are linked to each generator level to achieve iterative optimization and fusion of the feature encodings containing edge information.
- Frequency-sensitive image features are fed to the discriminator, and the generation quality of high-frequency edges and low-frequency regions of the image is improved by frequency discrimination.
- Extensive experiments demonstrate that our proposed method achieves state-of-the-art performance on the COCO-Stuff [CUF18] and Visual Genome [KZG*17] benchmarks.

2. Related Work

2.1. Conditional Image Generation

Generative Adversarial Networks (GANs) [GPAM*14] produces good outputs through game learning between the generator and discriminator. Nevertheless, the images generated in this way are random, and there is no control over which category the generated images fall into. To solve this problem, cGANs [MO14] came into being. The latter improves the former, allowing additional information to be input to the generator and discriminator as conditions for better control of the generation process. Specifically, conditions can be text descriptions [QZXT19b, LQLT19, RPG*21, YLS*19, QZXT19a, XZH*18], scene graphs [JGFF18, MAA*19, YTZC19], layouts [ZMYS19, MZS20, HLY*21, SW19, NFLW20], sketches [LWTT18, LQWL18, SLF*17], etc. We hope to get images that are more in line with our expectations by providing more conditional information to the model.

2.2. Image Generation from Layout

Compared with other conditions, the layout is simple, easy to be obtained, and the meaning expressed is clear, so it is favored by some researchers. In previous image generation tasks, layouts often acted as intermediate results or complementary features to the generated images.

Layout2Im [ZMYS19] is the first method to directly use a layout as an input source directly. Many studies decompose layout-based image generation tasks into multiple subtasks. LostGAN [SW19, SW20] transforms the layout-to-image problem into layout-to-mask-to-image to bridge the gap between the input layout and the synthesized image. While OC-GAN [SZB*20] is based on the layout-to-scene graph-to-image process, expecting to learn the relationship between objects through the scene graph. BachGAN [LCG*20] proposes that image generation can be carried out by separating foreground and background generation. In order to improve the generation quality at the object level, many methods provide their solutions. Attribute-guided layout2im [MZS20] takes the attribute information of the objects as an optional input condition for image generation. P-RaGAN [XLJL20] introduces a pair of relativistic average discriminators to solve the problem of object

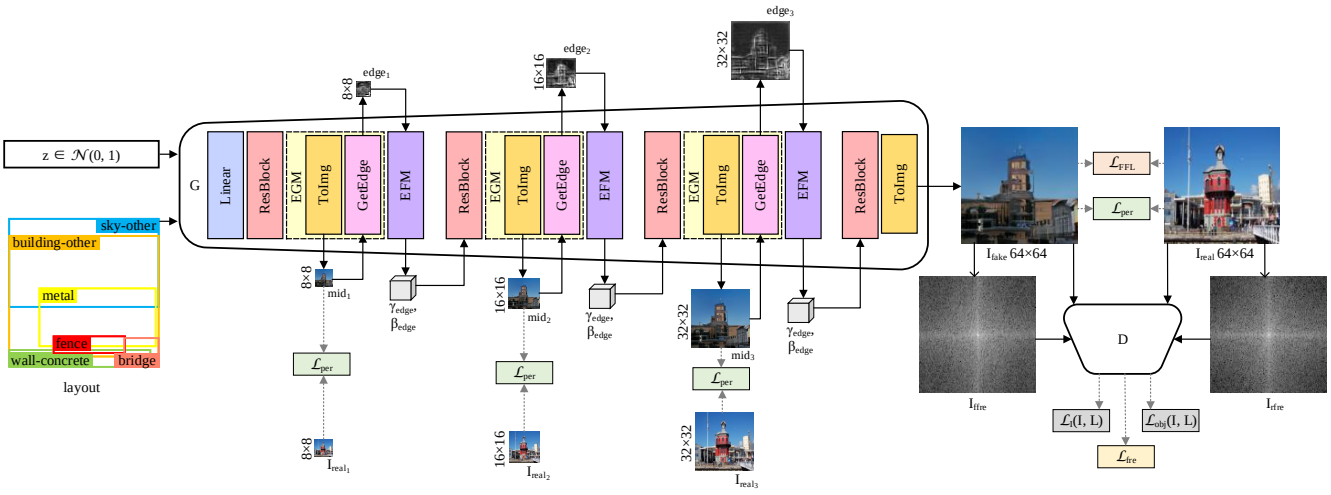

Figure 2: *The overview of our proposed EL-GAN (a version of final generated images with a resolution of* 64×64*). The inputs are a layout and a latent code z, where z is sampled from the normal distribution to characterize the diversity of the generated images. Edge generation module (EGM) and edge fusion module (EFM) are embedded in each generator level to perform image generation, edge information refinement, and fusion on the output of the multi-level features by ResBlocks.*

distortion and improve generative power. He *et al.* [HLY*21] introduces a context-aware feature transformation module and feeds location-sensitive image features into the discriminator to enhance the relationship between objects/stuff and their appearance. Li *et al.* [LWK*21] follows LostGAN's [SW19, SW20] idea and proposes the LAMA module, which is used to adapt overlapped or nearby object masks to help generate clean and semantically clear semantic masks, thereby improving the quality of image generation.

It can be seen that more and more researchers have begun to focus on the quality of individual objects in the image. However, there is almost no method to put forward practical solutions to the edge problem between individual components (objects, stuff, and background). This problem can seriously affect object recognizability and overall image quality.

3. Method

3.1. Overview

Our proposed EL-GAN architecture is shown in Figure 2. The edge generation module (EGM) and edge fusion module (EFM) are linked on each ResBlock (i.e., each level of the generator) to achieve iterative optimization and fusion of feature encodings containing edge information. In addition, the discriminator transforms the input image from the spatial domain to the frequency domain and then discriminates the image's authenticity based on the corresponding image features in these two domains.

3.2. Edge Generation Module (EGM)

We add an edge generation module (EGM) after each ResBlock to generate edge maps of different scales based on multi-level feature vectors. Due to the vast difference between feature vectors and edge maps, we decompose the task of generating edge maps from feature vectors into two stages. Specifically, the multi-level feature vectors are converted into intermediate image results of different scales through "ToImg." Then the edge maps of these images are extracted through "GetEdge."

ToImg is responsible for converting feature vectors to images. The output of the features by the generator is transformed into images of different scales using ToImg multiple times. Specifically, it converts the hierarchical feature x output by the i-th ResBlock into $N \times 3 \times H \times W$ images mid_i. The hierarchical feature x is a vector of shape $N \times C \times H \times W$, where N represents the mini-batch size, C represents the number of channels, H and W represent height and width, respectively. C, H, W depend on the level of ResBolck in the generator (in our experiments, the relationship between them and i is $C = 2^{11-i}$, $H = W = 8i$). The network structure of ToImg draws on the design of ToRGB in LostGAN-V2 [SW20], which is composed of "BatchNorm+ReLU+Conv3×3+Tanh," where Conv3×3 means a convolution layer with a kernel size of 3×3.

GetEdge is used to extract the edge maps of images. In order to enhance the edge information in the final generated image, we need to refine and fuse the edge information contained in the output of the hierarchical features by the generator. However, it is not intuitive and effective to directly refine such information from the multi-level features generated by ResBlock or the intermediate image results generated by ToImg. Therefore, we first extract edge maps from these images to facilitate subsequent extraction and fusion of edge feature information. It receives the images mid_i of different scales generated by ToImg and outputs the corresponding $N \times 1 \times H \times W$ edge maps $edge_i$ (see the grayscale images in Figure 2). We adopt and improve the holistically-nested edge detector (HED) [XT15] to achieve this task. The pre-trained VGG-16 [SZ15] is used as the image's feature extractor, and then it is connected to a convolutional layer after its 4th, 9th, 16th, 23rd,

and 30th layers (counting from 1) to obtain the edge map. Figure 3 shows the relationship between different layers of VGG-16 [SZ15] and the resulting edge maps. As seen from the figure, the higher the level of VGG-16 [SZ15], the more blurred the contour of the corresponding edge map. In order to obtain richer edge information, we only use the edge map corresponding to the fourth layer of VGG-16 [SZ15]. Like ToImg, the edge maps of different scales are obtained using GetEdge multiple times, and these edge maps contain different information to promote multi-scale learning.

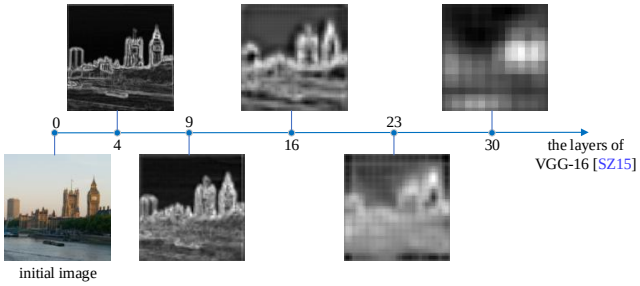

Figure 3: *Correspondence between different layers of VGG-16 [SZ15] and edge maps. The coordinate axis represents the layers of VGG-16 (counting from 1), and the grayscale image represents the edge map corresponding to a specific layer. In our experiments, we select the edge map corresponding to the fourth layer of VGG-16 with rich edge information.*

3.3. Edge Fusion Module (EFM)

The primary function of the edge fusion module (EFM) is to effectively refine the information in edge maps $edge_i$ to obtain feature encodings, then integrate these encodings containing edge information into the following image generation process by modulating the affine transformation parameters in the normalization layers. The detailed implementation of the edge fusion module (EFM) is shown in Figure 4(a). The following describes how the information in edge maps is refined and fused from the perspective of the normalization process. Normalization is generally divided into two steps: feature standardization and feature recalibration.

Feature Standardization. For the hierarchical feature x output by the ResBlock, we first normalize x by computing the channel-wise mean μ_c and variance σ_c^2 as done in the BatchNorm [IS15]. The formula is

$$\hat{x} = \frac{x - \mu_c}{\sqrt{\sigma_c^2 + \varepsilon}}, \tag{1}$$

where ε is a constant added for numerical stability. By standardizing the features, the adverse effects caused by singular data can be reduced, and the convergence speed of the model can be accelerated.

Feature Recalibration. In order to improve the representation ability of the model, the recalibration operation of features is essential. It is achieved by performing an affine transformation on the standardized features $\hat{x}$ with two parameters, γ_{edge} and β_{edge}.

In our task, the edge information in the final generated image is expected to be enhanced. So we first perform a Sigmoid operation on the edge map $edge_i$ to normalize the value of each pixel between 0 and 1. The closer the value of a pixel is to 1, the more likely the point is to be an edge. The normalized edge map is multiplied by the feature x output by ResBlock as a scale factor to obtain the feature x_{scale} containing edge information. In this way, the weight of non-edge pixel values in x is reduced, and the weight of edge pixel values in x is correspondingly enhanced. In this way, it is possible to explicitly focus on generating high-frequency information such as edge contours in the image.

The feature information in edge maps is high-level and abstract, and it is crucial to effectively refine the information in it. Inspired by SPADE [PLWZ19], we use a two-layer convolution operation to achieve the refinement of feature encodings. Specifically, the first layer of convolution projects the feature x_{scale} onto an embedding space and preliminarily refines the information in the edge map. The second layer of convolution refines this information again to obtain modulation parameters γ_{edge} and β_{edge} that vary with the spatial position. The parameters here refer to the feature encodings mentioned above that contain edge information. Finally, we use these two parameters to recalibrate the feature $\hat{x}$ to obtain $\widetilde{x}$. The formula is

$$\widetilde{x} = \gamma_{edge} \cdot \hat{x} + \beta_{edge}. \tag{2}$$

In addition, we also tried the implementation as shown in Figure 4(b). The convolution operation is used to directly refine the information in the edge map $edge_i$, and then the modulation parameters γ_{edge} and β_{edge} are obtained. After experimental verification, the effect of this setting is not good. For details, see w/o scale_factor of the ablation experiment in Sec. 4.7. We believe that the reason why this experimental setting is not good is that the feature extraction operation is performed directly on the edge map $edge_i$ with less information, and the information in the obtained feature x is not fully utilized. However, by using the edge map $edge_i$ as a scale factor to normalize the feature x and then using convolution to refine the information in it, the information of the edge map $edge_i$ and the feature x is fully utilized, which is a more effective way to play the role of the edge map.

3.4. Discriminator with Frequency Discrimination

In response to the lack of definition for image frequency information in the discriminator, we propose transforming the image from the spatial domain to the frequency domain. The discriminator is fed with frequency-sensitive image features, which are used to score the image's authenticity in the frequency domain.

In existing tasks of generating images from layouts, most discriminators first extract the features of the input image or the objects in it and then score their authenticity. However, this calculation process only pays attention to the intensity of the semantic features learned in the spatial domain, but not the intensity of the semantic features learned in the frequency domain, which causes the problem that the definition for image frequency information is missing in the discriminator.

To address this issue, we propose a frequency-sensitive loss item $\mathcal{L}_{fre}$ complementary to existing spatial domain losses. The details

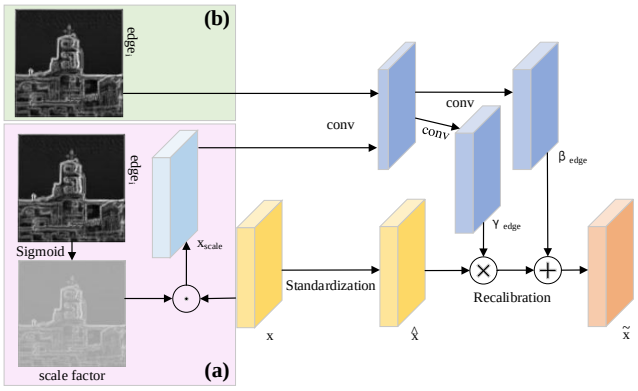

Figure 4: *Detailed implementation of the edge fusion module (EFM). (a) and (b) are two different ways, and the way (a) was chosen in our experiments. "conv" stands for the convolution operation.*

of this loss item can be found in Sec. 3.6, and here is the work that needs to be done before applying this loss item. Inspired by [JDWL21], we apply the 2D discrete Fourier transform to the image to get its frequency representation I_{fre}. The frequency value at each coordinate in the frequency domain depends on the pixel value at all coordinates in the spatial domain. The formula is expressed as follows:

$$F(u,v) = \sum_{x=0}^{W-1}\sum_{y=0}^{H-1} f(x,y)\cdot e^{-i2\pi(\frac{ux}{W}+\frac{vy}{H})}, \tag{3}$$

where W and H represent the width and height of the image; $f(x,y)$ represents the pixel value of the image in the spatial domain at (x,y) coordinate; $F(u,v)$ is the frequency value of the image in the frequency domain at (u,v) coordinate; e is the base of the natural logarithm, and i is the imaginary unit. According to Euler's formula as shown in Eq. (4):

$$e^{-i\theta} = cos\theta + isin\theta, \tag{4}$$

the natural exponential function in Eq. (3) can be rewritten as

$$e^{-i2\pi(\frac{ux}{W}+\frac{vy}{H})} = cos2\pi(\frac{ux}{W}+\frac{vy}{H}) - isin2\pi(\frac{ux}{W}+\frac{vy}{H}). \tag{5}$$

Therefore, the frequency value of the image in the frequency domain is calculated by the orthogonal cosine and sine functions, which represent the real and imaginary parts of the frequency value, respectively.

We apply convolutional neural networks to the frequency representation of an image to extract image features at the frequency level. The discriminator is asked to rate the image's authenticity by this feature information. High frequencies correspond to places in the image where the grayscale or brightness change drastically, that is, edge contours. Low frequencies correspond to places in the image where the grayscale and brightness changes gently, that is, large areas. By discriminating images in the frequency domain, it helps the generator to generate correct high-frequency and low-frequency information, corresponding to higher-quality edge contours and large areas of the spatial domain image.

Additionally, we use a projection discriminator, inspired by projection-based cGANs [MK18], when scoring the authenticity of objects contained in the spatial domain image. The word embedding of label l in the layout is added into the discriminator as additional condition information, participating in the computation of the adversarial hinge loss for objects in the image. The discriminator and the generator have been fighting against and promoting each other, making the generated images and objects in them better and better.

3.5. Generator

The generator inputs have two parts: a layout and a latent code z. The category label l of each object/stuff in the layout is transformed into a word embedding with dimension d_l ($d_l = \mathbf{180}$ in our experiments). The latent code z is a vector of dimension d_z sampled from the normal distribution to characterize the diversity of the generated images ($d_z = \mathbf{128}$ in our experiments). The generator uses ResNet [HZRS16] as the backbone architecture, containing a linear layer and multiple ResBlocks. We use B ResBlocks to generate final images with resolutions of $4^{B-1}\times 4^{B-1}$ (e.g., 4 ResBlocks can generate images with a resolution of 64×64).

3.6. The Loss Functions

Frequency-sensitive Loss $\mathcal{L}_{fre}$. This loss term penalizes images not correctly classified in the frequency domain and is complementary to the existing spatial domain losses. Inspired by the hinged version of the traditional adversarial loss [TRB17], the $\mathcal{L}_{fre}$ loss item has different forms in the generator and discriminator. In the discriminator, $\mathcal{L}_{fre}$ is formulated as follows:

$$\mathcal{L}_{fre}(I_{fre}) = \begin{cases} max(0, 1-p_{I_{fre}}), & \text{If } I_{fre} \text{ means } I_{rfre} \\ max(0, 1+p_{I_{fre}}), & \text{If } I_{fre} \text{ means } I_{ffre} \end{cases} \tag{6}$$

In the generator, $\mathcal{L}_{fre}$ is formulated as follows:

$$\mathcal{L}_{fre}(I_{ffre}) = -p_{I_{ffre}}. \tag{7}$$

Among them, I_{rfre} represents the frequency representation corresponding to the ground truth I_{real}; I_{ffre} represents the frequency representation corresponding to the final generated image I_{fake}; p_* represents the true score of $*$ output by the network layer.

Focal Frequency Loss $\mathcal{L}_{FFL}$. This loss item considers the difference between the final generated image I_{fake} and the ground truth I_{real} in the frequency domain and improves the quality of the generated image by reducing this difference. So we introduce this loss item to better handle the high-frequency information in the image generation process, that is, edge contours. The loss item is expressed as the following formula:

$$\mathcal{L}_{FFL} = FFL(I_{real}, I_{fake}). \tag{8}$$

Perceptual Loss $\mathcal{L}_{per}$. This loss item constrains the distance between the generated and ground truth images in deep-level features. We add this loss item between the intermediate images mid_i output

by the i-th ToImg and the ground truth images of the corresponding resolution. The formula is

$$\mathcal{L}_{per_i} = Dis(F(I_{real_i}), F(mid_i)), \tag{9}$$

where Dis is the distance function, and F represents the network that extracts the features of the images (in our experiments, Dis is the L1 distance, and F is the VGG-19 [SZ15] network). We need to scale the ground truth images to get images $Ireal_i$ of the same size as the mid_i. Note that when i refers to the last ToImg in the generator, I_{real_i} refers to I_{real}, and mid_i refers to I_{fake}.

Adversarial Loss $\mathcal{L}_t(I,L)$**.** Unlike the traditional adversarial loss [GPAM*14], we adopt its hinge version [TRB17]. Its general idea is to make the distance between images/objects that are not correctly classified and images/objects correctly classified sufficiently far away. The adversarial loss in the discriminator is expressed by the formula as follows:

$$\mathcal{L}_t(I,L) = \begin{cases} max(0, 1-p_t), & \text{if I is a real image} \\ max(0, 1+p_t), & \text{if I is a fake image} \end{cases} \tag{10}$$

The adversarial loss in the generator is expressed as follows:

$$\mathcal{L}_t(I,L) = -p_t, \text{if I is a fake image} \tag{11}$$

where $t \in \{I, obj_1, ..., obj_m\}$ and p_t represent the realness score of t output by the network layer. The subscript m represents the number of objects in an image. We use $L_I(I,L)$ to represent the adversarial loss of the image, that is, $t \in \{I\}$. Similarly, we use $L_{obj}(I,L)$ to represent the objects' adversarial loss, that is, $t \in \{obj_1, ..., obj_m\}$.

Total Loss. In general, the total loss function of the discriminator is expressed as:

$$\mathcal{L}_D = \lambda_{fre}\mathcal{L}_{fre} + \lambda_I\mathcal{L}_I(I,L) + \lambda_{obj}\mathcal{L}_{obj}(I,L). \tag{12}$$

The total loss function of the generator is expressed as:

$$\begin{aligned} \mathcal{L}_G =& \lambda_{fre}\mathcal{L}_{fre} + \lambda_{FFL}\mathcal{L}_{FFL} \\ &+ \lambda_{per}\sum_{i=1}^{n}\mathcal{L}_{per_i} + \lambda_I\mathcal{L}_I(I,L) + \lambda_{obj}\mathcal{L}_{obj}(I,L), \end{aligned} \tag{13}$$

where n is the number of ResBlocks in the generator. $\mathcal{L}_{fre}$, $\mathcal{L}_I(I,L)$, and $\mathcal{L}_{obj}(I,L)$ have different forms for the discriminator and generator (see Eq. (6), Eq. (7), Eq. (10), and Eq. (11) for details). $\lambda*$ indicates the proportion of the following loss item to the total loss. We can get better results by adequately adjusting the proportion of each loss item.

4. Experiments

4.1. Datasets

We used the two most commonly used benchmarks, the COCO-Stuff [CUF18] and Visual Genome (VG) [KZG*17] datasets. The annotation file of the COCO-Stuff dataset contains 80 *object instance* categories (such as person, bicycle, etc.) and 91 *stuff* categories (such as clouds, fog, etc.). Following the settings of [SW19, SW20], objects whose bounding box takes up less than 2% of the image will be ignored, and the number of objects in each image is between 3 and 8. 74777 and 3097 images are used for training and testing, respectively. The Visual Genome dataset contains annotation information for 178 object categories. We follow the settings of [SW19, SW20] to preprocess the dataset, using 62565 and 5062 images containing 3 to 30 objects for model training and testing.

4.2. Implementation Details

Our algorithm is implemented with PyTorch [PGM*19]. We use Synchronized Batch Normalization [IS15] in each ResBlock of the generator. It can make full use of existing computing resources to achieve synchronization. This operation is performed on all devices instead of independently limiting to each GPU. It is also used in the edge fusion module (EFM). Spectral Normalization [MKKY18] is used in both the generator and discriminator.

The generator and discriminator use Adam [KB15] as the optimizer, and the learning rate is set to $1e^{-4}$. The batch size is 16, and a total of 200 epochs are trained. We use 4 Tesla V100 GPUs for our experiments. The weight parameters λ_{FFL}, λ_{per}, λ_I, λ_{obj} are set to 1, 1, 0.1, 1, respectively. And λ_{fre} is 1 when facing I_{rfre} and 0.1 when facing I_{ffre}.

4.3. Methods in Comparison

We compared with five previous methods: Layout2Im [ZMYS19], attribute-guide layout2im [MZS20], LostGAN-V2 [SW20], He *et al.* [HLY*21] and Li *et al.* [LWK*21]. For brevity of description and neatness of presentation in figures or tables, we abbreviate the model proposed by He *et al.* [HLY*21] as CAL2I. The attribute-guide layout2im [MZS20] model is abbreviated as AG-layout2im and the model proposed by Li *et al.* [LWK*21] as LAMA.

Layout2Im [ZMYS19] pioneered the generation of images using layouts directly as input sources. It divides the representation of each object in the image into a specified/certain part (category) and an unspecified/uncertain part (appearance). It first builds a feature map for each object in the input layout. Then the convolutional Long-Short-Term Memory (cLSTM) network is used to fuse the feature maps of the individual objects into a hidden feature map, which is used to generate the final image.

AG-layout2im [MZS20] is an extension of Layout2Im [ZMYS19]. In addition to layout information, it also adds attributes of each object as optional input, where attributes can be specified by the user or sampled from prior class distributions of object-attribute co-occurrence counts. It learns 106 (/40000) attributes in the dataset, enhancing the ability to semantically control individual object instances. At the same time, it introduces SPADE [PLWZ19] to prevent semantic information from being washed away by conventional normalization layers.

LostGAN-V2 [SW20] learns fine-grained mask maps in a weakly-supervised manner during image generation to bridge the gap between layouts and images. It is equivalent to transforming the layout-to-image problem into a layout-to-mask-to-image problem. Its input includes spatial layouts and style codes, and style control includes two levels of image and mask.

CAL2I [HLY*21] introduces a context-aware feature transformation module in the generator. Each object and stuff update their respective features by checking other existing objects or stuff in the image through the self-attention mechanism. At the same time,

Table 1: *Comparison results on the COCO-Stuff (COCO) and Visual Genome (VG) datasets. Bold means the best performance under the same resolution conditions in the same column.*

Methods	Resolution	IS↑		FID↓		DS↑	
		COCO	VG	COCO	VG	COCO	VG
Real images	64 × 64	12.58 ± 0.59	11.81 ± 0.55	-	-	-	-
Real images	128 × 128	20.57 ± 1.11	19.67 ± 0.88	-	-	-	-
Real images	256 × 256	27.71 ± 1.64	26.78 ± 0.95	-	-	-	-
Layout2Im [ZMYS19]	64 × 64	8.59 ± 0.39	7.71 ± 0.40	43.01	38.78	0.15 ± 0.07	0.17 ± 0.09
AG-layout2im [MZS20]	64 × 64	-	8.1 ± 0.2	-	33.09	-	0.10 ± 0.02 0.20 ± 0.01
LAMA [LWK*21]	64 × 64	8.98 ± 0.63	7.64 ± 0.34	28.48	22.70	**0.37 ± 0.10**	**0.38 ± 0.09**
Ours64	64 × 64	**10.52 ± 0.62**	**8.52 ± 0.51**	**24.25**	**19.27**	0.17 ± 0.08	0.16 ± 0.08
AG-layout2im [MZS20]	128 × 128	-	8.5 ± 0.1	-	39.12	-	0.15 ± 0.09
LostGAN-V2 [SW20]	128 × 128	12.58 ± 0.79	10.14 ± 0.45	31.12	28.97	0.45 ± 0.09	0.42 ± 0.09
CAL2I [HLY*21]	128 × 128	13.52 ± 0.48	11.25 ± 0.50	30.40	25.67	0.16 ± 0.07	0.30 ± 0.10
LAMA [LWK*21]	128 × 128	13.02 ± 0.58	10.45 ± 0.48	31.06	27.27	**0.46 ± 0.09**	**0.49 ± 0.09**
Ours128	128 × 128	**13.88 ± 0.53**	**12.05 ± 0.35**	**29.55**	**20.95**	0.35 ± 0.10	0.27 ± 0.08
LostGAN-V2 [SW20]	256 × 256	15.86 ± 0.86	**13.38 ± 0.53**	37.61	**32.38**	**0.55 ± 0.09**	0.53 ± 0.10
LAMA [LWK*21]	256 × 256	17.06 ± 1.04	12.96 ± 0.52	38.08	36.92	0.49 ± 0.11	**0.54 ± 0.10**
Ours256	256 × 256	**17.24 ± 0.92**	13.26 ± 0.76	**36.49**	35.72	0.44 ± 0.10	0.42 ± 0.10

it uses the Gram matrix computed from the feature maps of the generated object images to preserve location-sensitive information in the discriminator, which greatly enhances the appearance of objects.

LAMA [LWK*21] assumes that it is important to generate clean and semantically clear semantic masks in the layout-to-image process. Based on this assumption, the Locality-Aware Mask Adaption (LAMA) module is proposed. It adapts the raw semantic mask to a cleaner one by scaling the mask values of each object in each pixel individually with a learned matching mechanism.

4.4. Evaluation Metrics

We perform quantitative, qualitative, and ablation studies on the models. We hope that the generated images conform to the size and position information of the objects provided in the layout. They should also look clear, real, and diverse. We use three metrics to quantitatively evaluate the images generated by the model: Inception Score (IS) [SGZ*16], Fréchet Inception Distance (FID) [HRU*17], and Diversity Score (DS).

Inception Score (IS) [SGZ*16] mainly focuses on the clarity and diversity of images. It uses the Inception-V3 [SVI*16] network pre-trained on the ImageNet [DDS*09] dataset for calculation. The diversity in IS refers to the diversity of the categories of objects in the image. For example, if a model can generate sufficiently diverse pictures, the distribution of pictures generated by the model with a high IS score in each category should be even. The higher the IS value, the better the image quality.

Fréchet Inception Distance (FID) [HRU*17] focuses on the distance between the generated and ground truth image in the feature space. It first uses the Inception-V3 [SVI*16] network to perform feature extraction on the image, then uses the Gaussian model to model the feature space, and finally uses the mean and covariance matrix to calculate the distance between the two distributions. The lower the FID value, the better the quality of the generated image.

Diversity Score (DS) compares the perceptual similarity of two images generated from the same layout in the deep feature space. Unlike the diversity in IS, DS focuses on the diversity of two images corresponding to the same layout, that is, the degree to which one of the generator's inputs z works. Specifically, we use the LPIPS metric [ZIE*18] to calculate DS, where we use AlexNet [KSH12] to extract the deep features of the image.

4.5. Quantitative Results

Table 1 shows the comparison results of the quantitative metrics. The data in the table are calculated using the pre-trained models provided by the relevant authors under their respective datasets division and processing conditions, except for AG-layout2im [MZS20]. Because we did not find its pre-trained model, we reused the data provided in its paper. We denote our models that generate images with resolutions of 64 × 64, 128 × 128 and 256 × 256 as Ours64, Ours128 and Ours256, respectively.

For IS and FID metrics, our model outperforms all other methods except Ours256 is slightly inferior to LostGAN-V2 [SW20] on the VG dataset. It is worth noting that at a resolution of 64 × 64, the FID metric of our model under the COCO dataset is more than 4 points lower than the second-place LAMA [LWK*21]. Similarly, under the condition of 128 × 128 resolution, it is nearly 5 points lower than the second-place CAL2I [HLY*21] on the VG dataset. On the DS metric, although the performance of our model is not the best, the quality of images and the edge contours of objects generated by our model have great advantages over other models, see Sec. 4.6 for details.

Figure 5: *Examples of generating samples from a given layout by different methods. The left side of the dashed line is the results under COCO-Stuff [CUF18], while the right is the sampling results under the Visual Genome [KZG*17] dataset. Only Layout2Im [ZMYS19], LAMA64 [LWK*21], and Ours64 images have a resolution of* 64×64*, and the rest are* 128×128*.*

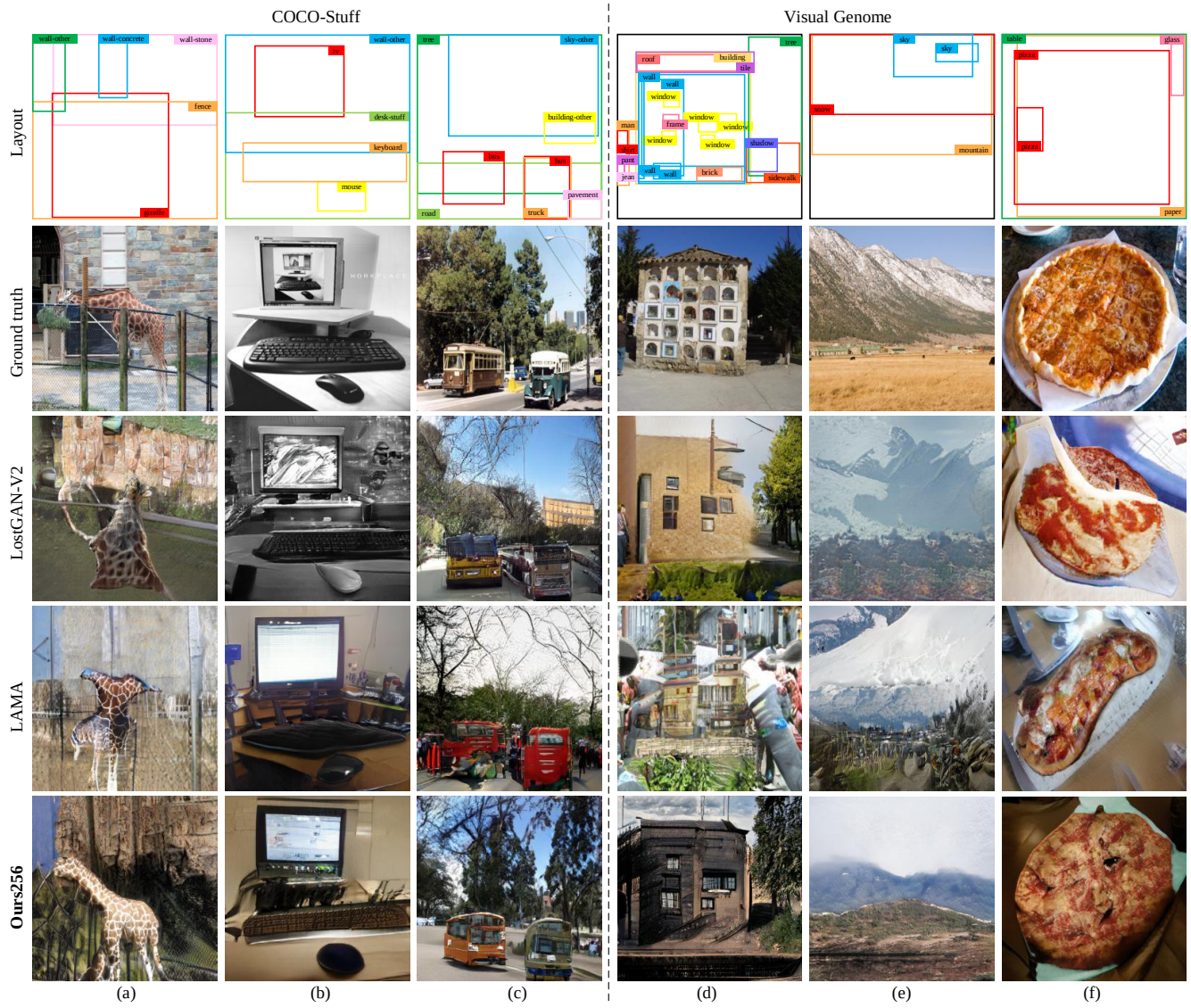

Figure 6: *Examples of generating samples from a given layout by different methods. The left side of the dashed line is the results under COCO-Stuff [CUF18], while the right side is the sampling results under the Visual Genome [KZG*17] dataset. All images are of* 256×256 *resolution.*

4.6. Qualitative Results

Figures 5 and 6 show images generated by our method and some other methods, including Layout2Im [ZMYS19], LAMA [LWK*21], LostGAN-V2 [SW20], and CAL2I [HLY*21]. Also, more comparisons are provided in the supplemental material. As can be seen from these examples, the generative ability of our model is impressive, especially in the processing of edge contours.

At a resolution of 64×64, it is difficult for Layout2Im and LAMA64 to generate recognizable objects. Nevertheless, the objects generated by Ours64 are plausible and reasonable. The edges of objects in (a), (b), (e), and (h) are smooth and full, which is in line with the human visual system.

Under the condition of 128×128 resolution, the lines of the *vase* generated by LostGAN-V2, CAL2I, and LAMA128 in (b) are distorted, which obviously does not conform to people's perception of vases in the real world. In (c), the legs of the *zebra* generated by CAL2I are missing, and LAMA128 fails to generate the *zebra* object. The *boat* and *train* serve as central subject objects in (f) and (h), and the remaining methods generate these two objects with poor recognizability, while Ours128 generates the *boat* and *train* with high quality. The layout of (g) is complex and contains many objects. The images generated by LostGAN-V2 and LAMA128 look cluttered. While the image generated by Ours128 in (g) is clean and straightforward, and the edge lines of objects such as *lamps* are apparent.

Figure 7: *Qualitative comparison of ablation experiments on COCO-Stuff. All images are of* 128×128 *resolution. Areas requiring attention are marked with red dashed rectangles. See text for more details.*

Under the condition of 256×256 resolution, problems such as distortion and unreasonable edges of objects are easier to find. The *giraffe* generated by LostGAN-V2 and LAMA in (a) has only textures but no edge contours, which looks a bit scary. The right boundary of the *building* in (d) generated by LostGAN-V2 appears to blend with other component pixels, and the *paper* in (f) passes through the *pizza*, which is not in line with the actual situation. Most of the objects generated by LAMA are distorted, such as the *keyboard* in (b), the *bus* in (c), the *building* in (d), and the *pizza* in (f). Our model sometimes does not handle image details well enough, such as undesired cluttered pixels in the *desk-stuff* in (b), and an almost missing *man* in (d). But in comparison, the images generated by Ours256 look more natural and harmonious because the objects' edges are smoother and fuller.

4.7. Ablation Study

In this experiment, we verify the necessity of its existence by removing key components in our model. The experiment was completed on the COCO-Stuff [CUF18] dataset. The ablation experiment design is as follows:

w/o edge. Edge-related modules, specifically the edge generation module (EGM) and edge fusion module (EFM), are no longer used in the generator. This experiment verifies the effect of these two modules on image quality.

w/o scale_factor. In the edge fusion module (EFM), the edge map is no longer used as a scale factor to normalize the output of the features by ResBlock (see Figure 4(a)). Instead, the convolution operation is directly applied to the edge map to extract feature information (see Figure 4(b)). This experiment compares which of the two ways of adding edge information is more efficient.

w/o D_{fre}. Images are not converted to frequency representation in the discriminator. The $\mathcal{L}_{fre}$ loss item is also not used in the generator and discriminator during training. This experiment is to verify the effectiveness of frequency discrimination.

w/o $\mathcal{L}_{FFL}$. The $\mathcal{L}_{FFL}$ loss item is not used during training to see its effect on image quality.

The quantitative results are shown in Table 2. The Full model achieves the best results on IS and FID metrics. All models with individual key components removed show a drop in performance. Among them, the model that removes edge-related modules (w/o edge) has the worst effect, so adding edge information during image generation can effectively improve the quality of image generation. Compared with w/o edge, the effect of w/o scale_factor with edge information is improved. We can also see a gap still compared with the full model. So the way of adding edge information is crucial. Both w/o D_{fre} and w/o $\mathcal{L}_{FFL}$ degrade the performance of the model, too. Therefore, all these key components are essential to improving the generated image quality.

Table 2: *Ablation study of our model on COCO-Stuff. Bold indicates the top performer in the same column.*

Methods	IS↑	FID↓
w/o edge	12.63 ± 0.48	32.54
w/o scale factor	13.17 ± 0.51	31.27
w/o D_{fre}	13.74 ± 0.73	30.10
w/o $\mathcal{L}_{FFL}$	13.42 ± 0.65	30.25
Full model	$\mathbf{13.88 \pm 0.53}$	**29.55**

The qualitative results are shown in Figure 7. From these examples, it can be seen that most of the objects generated by w/o edge have problems of poor recognizability. The *cat* in (a), the two *elephants* in (c), the *zebra* in (d), and the *bus* in (e) do not see their original shape at all. Moreover, pixel blending occurs between adjacent *donuts* in (f). Compared with w/o edge, w/o scale_factor of adding edge information has a certain improvement in image quality. For example, the lines of the *zebra* in (d) and the *donuts* in (f) are clearer and smoother. But some objects are still difficult to identify, such as the *cat* in (a) and the *bus* in (e). w/o D_{fre} and w/o $\mathcal{L}_{FFL}$ use a scale factor operation for edge maps. It can be seen that the quality of the *elephants* generated in (c) and the *bus* in (e) are significantly improved by both methods compared to w/o scale_factor. But there are still some problems with the images they generate, such as the *cat* in (a), the *zebra* in (d), and the *donuts* in (f) do not work well. The full model fuses the advantages of each component to generate more recognizable objects and higher-quality images, as in (g) the *vase* looks plausible.

5. Conclusion

This paper proposes a novel Generative Adversarial Network architecture with enhanced edge information for layout-based image generation (EL-GAN). EGM and EFM are included in each level of the generator, which can effectively iteratively learn and optimize the feature encodings containing edge information through multi-scale learning. These feature encodings influence the subsequent image generation process by modulating the affine transformation parameters in the normalization layers. The purpose is to make the edge contours in the generated images clearer and solve the problems of distortion and poor recognizability of object-level generation. At the same time, frequency-sensitive image features are added to the discriminator, which enhances the definition of image frequency information. Extensive experiments verify the necessity of key components in our model and demonstrate that our proposed method achieves state-of-the-art performance. Our future work will continue to focus on improving object-level generation quality and implementing more general strategies that do not rely on specific model design.

Acknowledgment This work is supported in part by the National Key R&D Program of China (Grant no. 2021YFC3300205).

References

[CUF18] Caesar H., Uijlings J., Ferrari V.: Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (2018), pp. 1209–1218. 2, 6, 8, 9, 11

[DDS*09] Deng J., Dong W., Socher R., Li L.-J., Li K., Fei-Fei L.: Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255. 7

[GPAM*14] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y.: Generative adversarial nets. *Advances in neural information processing systems (NIPS) 27* (2014), 2672–2680. 2, 6

[HLY*21] He S., Liao W., Yang M. Y., Yang Y., Song Y.-Z., Rosenhahn B., Xiang T.: Context-aware layout to image generation with enhanced object appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 15049–15058. 2, 3, 6, 7, 9

[HRU*17] Heusel M., Ramsauer H., Unterthiner T., Nessler B., Hochreiter S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems (NIPS) 30* (2017), 6626–6637. 7

[HZRS16] He K., Zhang X., Ren S., Sun J.: Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (2016), pp. 770–778. 5

[IS15] IOFFE S., SZEGEDY C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning (ICML)* (2015), PMLR, pp. 448–456. 4, 6

[JDWL21] JIANG L., DAI B., WU W., LOY C. C.: Focal frequency loss for image reconstruction and synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 13919–13929. 5

[JGFF18] JOHNSON J., GUPTA A., FEI-FEI L.: Image generation from scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (2018), pp. 1219–1228. 1, 2

[KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)* (2015). 6

[KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems (NIPS) 25* (2012). 7

[KZG*17] KRISHNA R., ZHU Y., GROTH O., JOHNSON J., HATA K., KRAVITZ J., CHEN S., KALANTIDIS Y., LI L.-J., SHAMMA D. A., ET AL.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision (IJCV) 123*, 1 (2017), 32–73. 2, 6, 8, 9

[LCG*20] LI Y., CHENG Y., GAN Z., YU L., WANG L., LIU J.: Bachgan: High-resolution image synthesis from salient object layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 8365–8374. 2

[LQLT19] LI B., QI X., LUKASIEWICZ T., TORR P. H.: Controllable text-to-image generation. *Conference and Workshop on Neural Information Processing Systems (NIPS)* (2019). 2

[LQWL18] LIU Y., QIN Z., WAN T., LUO Z.: Auto-painter: Cartoon image generation from sketch by using conditional wasserstein generative adversarial networks. *Neurocomputing 311* (2018), 78–87. 2

[LWK*21] LI Z., WU J., KOH I., TANG Y., SUN L.: Image synthesis from layout with locality-aware mask adaption. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 13819–13828. 3, 6, 7, 8, 9

[LWTT18] LU Y., WU S., TAI Y.-W., TANG C.-K.: Image generation from sketch constraint using contextual gan. In *Proceedings of the European conference on computer vision (ECCV)* (2018), pp. 205–220. 1, 2

[MAA*19] MITTAL G., AGRAWAL S., AGARWAL A., MEHTA S., MARWAH T.: Interactive image generation using scene graphs. *arXiv preprint arXiv:1905.03743* (2019). 2

[MK18] MIYATO T., KOYAMA M.: cgans with projection discriminator. *International Conference on Learning Representations (ICLR)* (2018). 5

[MKKY18] MIYATO T., KATAOKA T., KOYAMA M., YOSHIDA Y.: Spectral normalization for generative adversarial networks. *International Conference on Learning Representations (ICLR)* (2018). 6

[MO14] MIRZA M., OSINDERO S.: Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014). 2

[MZS20] MA K., ZHAO B., SIGAL L.: Attribute-guided image generation from layout. *British Machine Vision Conference (BMVC)* (2020). 1, 2, 6, 7

[NFLW20] NIU T., FENG F., LI L., WANG X.: Image synthesis from locally related texts. In *Proceedings of the 2020 International Conference on Multimedia Retrieval (ICMR)* (2020), pp. 145–153. 1, 2

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., ET AL.: Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems 32* (2019). 6

[PLWZ19] PARK T., LIU M.-Y., WANG T.-C., ZHU J.-Y.: Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 2337–2346. 4, 6

[QZXT19a] QIAO T., ZHANG J., XU D., TAO D.: Learn, imagine and create: Text-to-image generation from prior knowledge. *Advances in Neural Information Processing Systems (NIPS) 32* (2019), 887–897. 2

[QZXT19b] QIAO T., ZHANG J., XU D., TAO D.: Mirrorgan: Learning text-to-image generation by redescription. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 1505–1514. 1, 2

[RPG*21] RAMESH A., PAVLOV M., GOH G., GRAY S., VOSS C., RADFORD A., CHEN M., SUTSKEVER I.: Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092* (2021). 2

[SGZ*16] SALIMANS T., GOODFELLOW I., ZAREMBA W., CHEUNG V., RADFORD A., CHEN X.: Improved techniques for training gans. *Advances in neural information processing systems (NIPS) 29* (2016), 2234–2242. 7

[SLF*17] SANGKLOY P., LU J., FANG C., YU F., HAYS J.: Scribbler: Controlling deep image synthesis with sketch and color. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5400–5409. 2

[SVI*16] SZEGEDY C., VANHOUCKE V., IOFFE S., SHLENS J., WOJNA Z.: Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2818–2826. 7

[SW19] SUN W., WU T.: Image synthesis from reconfigurable layout and style. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 10531–10540. 2, 3, 6

[SW20] SUN W., WU T.: Learning layout and style reconfigurable gans for controllable image synthesis. *IEEE Trans on Pattern Analysis and Machine Intelligence (TPAMI)* (2020). 1, 2, 3, 6, 7, 9

[SZ15] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)* (2015). 3, 4, 6

[SZB*20] SYLVAIN T., ZHANG P., BENGIO Y., HJELM R. D., SHARMA S.: Object-centric image generation from layouts. *arXiv preprint arXiv:2003.07449 1*, 2 (2020), 4. 1, 2

[TRB17] TRAN D., RANGANATH R., BLEI D. M.: Hierarchical implicit models and likelihood-free variational inference. *Advances in Neural Information Processing Systems (NIPS)* (2017), 5523–5533. 5, 6

[XLJL20] XU T., LIU K., JI Y., LIU C.: Image generation from layout via pair-wise ragan. In *International Conference on Neural Computing for Advanced Applications (NVAA)* (2020), Springer, pp. 193–206. 2

[XT15] XIE S., TU Z.: Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision (ICCV)* (2015), pp. 1395–1403. 3

[XZH*18] XU T., ZHANG P., HUANG Q., ZHANG H., GAN Z., HUANG X., HE X.: Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (2018), pp. 1316–1324. 2

[YLS*19] YIN G., LIU B., SHENG L., YU N., WANG X., SHAO J.: Semantics disentangling for text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 2327–2336. 2

[YTZC19] YANG X., TANG K., ZHANG H., CAI J.: Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 10685–10694. 2

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 586–595. 7

[ZMYS19] ZHAO B., MENG L., YIN W., SIGAL L.: Image generation from layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 8584–8593. 1, 2, 6, 7, 8, 9